

**Universidad Nacional Autónoma De México**  
**Facultad de Estudios Superiores Aragón**  
**Ingeniería en Computación**

**Alumno: Yahir Adrian Ruiz Machuca**

**Profesor: Aaron Velasco Agustín.**

**Grupo: 1710.**

**Semestre: 2025 – 1.**

**Fecha de entrega: 27 – Noviembre – 2024.**

## Visualización de index (Página principal).

The screenshot shows a web browser window titled "CRUD (Proyecto Final)" with the URL "localhost:1234". The main content area is titled "Proyecto Final (CRUD)" and contains a table with four rows of data:

ID	NOMBRE	CORREO	UNIVERSIDAD	CUENTA	OPCIONES
1	YAHIR ADRIAN RUIZ MACHUCA	yahiradrianruiz10@gmail.com.mx	UNAM	319123456	<a href="#">Editar</a>   <a href="#">Eliminar</a>
3	DIANA JARAMILLO LOPEZ PALACIOS	dianalopez27@outlook.com.mx	CU (FACULTAD DE INGENIERIA)	319168746	<a href="#">Editar</a>   <a href="#">Eliminar</a>
4	LUIS ANGEL RAMIREZ JUAREZ	luisjuarez@gmail.com.mx	FES ARAGÓN (UNAM)	319876543	<a href="#">Editar</a>   <a href="#">Eliminar</a>

The status bar at the bottom shows system information: "ÚLTIMA HORA Trump nombra a...", "Buscar", "04:11 p. m.", and "27/11/2024".

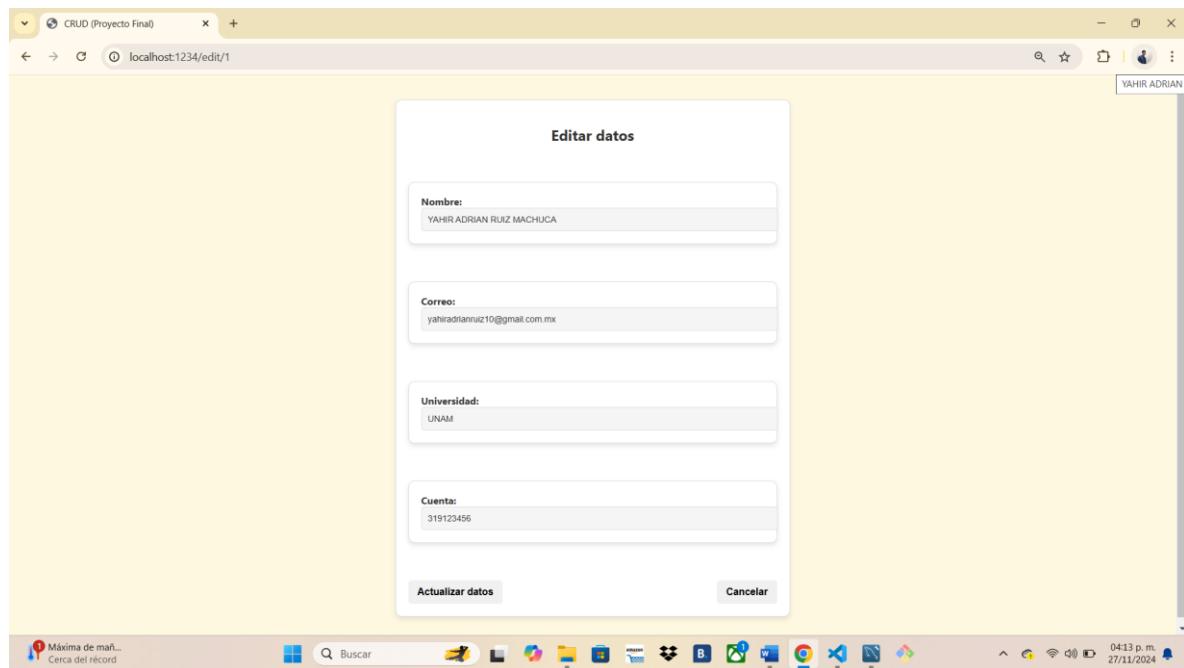
## Interfaz para agregar un nuevo usuario.

The screenshot shows a web browser window titled "CRUD (Proyecto Final)" with the URL "localhost:1234/agregar". The main content area is titled "Agregar usuario" and contains four input fields:

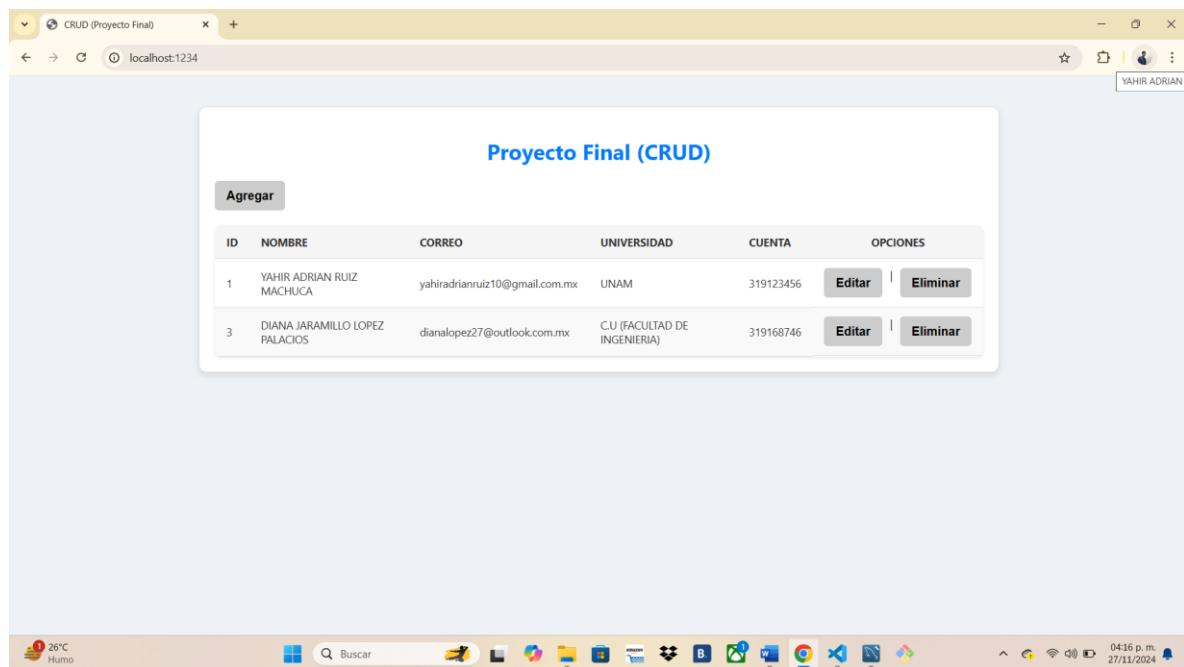
- Nombre: (Name) - An input field for entering the name.
- Correo: (Email) - An input field for entering the email address.
- Universidad: (University) - An input field for entering the university.
- Cuenta: (Account) - An input field for entering the account number.

At the bottom of the form are two buttons: "Registrar" (Register) and "Cancelar" (Cancel). The status bar at the bottom shows system information: "Máxima de mañ... Cerca del récord", "Buscar", "04:12 p. m.", and "27/11/2024".

Interfaz para poder editar datos del usuario.



Eliminación de usuario a través del botón Eliminar.



## Observación de los usuarios registrados antes de eliminar al último usuario de la Base de Datos.

```

CREATE DATABASE IF NOT EXISTS crud;
USE crud;

CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT NOT NULL, -- ID del usuario
    name VARCHAR(70) NOT NULL, -- Nombre del usuario
    email VARCHAR(100) NOT NULL, -- Correo electrónico del usuario
    universidad VARCHAR(100) NOT NULL, -- Universidad del usuario
    cuenta INT(9) NOT NULL, -- Número de cuenta de nuestro usuario
    PRIMARY KEY (id) -- Definimos la clave primaria como 'id'
);

INSERT INTO users (name, email, universidad, cuenta) VALUES ('YAHIR ADRIAN RUIZ MACHUCA', 'yahiradrianruiz10@gmail.com.mx', 'UNAM', '319123456');
INSERT INTO users (name, email, universidad, cuenta) VALUES ('DIANA JARAMILLO LOPEZ PALACIOS', 'dianalopez27@outlook.com.mx', 'C.U (FACULTAD DE INGENIERIA)', '319168746');
INSERT INTO users (name, email, universidad, cuenta) VALUES ('LUIS ANGEL RAMIREZ JUAREZ', 'lusjarez@gmail.com.mx', 'FES ARAGON (UNAM)', '319876543');

SELECT * FROM users;

```

Result Grid:

	id	name	email	universidad	cuenta
1	1	YAHIR ADRIAN RUIZ MACHUCA	yahiradrianruiz10@gmail.com.mx	UNAM	319123456
2	3	DIANA JARAMILLO LOPEZ PALACIOS	dianalopez27@outlook.com.mx	C.U (FACULTAD DE INGENIERIA)	319168746
3	4	LUIS ANGEL RAMIREZ JUAREZ	lusjarez@gmail.com.mx	FES ARAGON (UNAM)	319876543
4	5				

Action Output:

#	Time	Action	Message	Duration / Fetch
7	14:25:13	INSERT INTO users (name, email, universidad, cuenta) VALUES ('YAHIR ADRIAN RUIZ MACHUCA', 'yahiradrianruiz10@gmail.com.mx', 'UNAM', '319123456');	1 row(s) affected	0.016 sec
8	14:25:17	SELECT * FROM users LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
9	15:46:36	SELECT * FROM users LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

## Observación de los usuarios registrados después de eliminar al último usuario de la Base de Datos.

```

CREATE DATABASE IF NOT EXISTS crud;
USE crud;

CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT NOT NULL, -- ID del usuario
    name VARCHAR(70) NOT NULL, -- Nombre del usuario
    email VARCHAR(100) NOT NULL, -- Correo electrónico del usuario
    universidad VARCHAR(100) NOT NULL, -- Universidad del usuario
    cuenta INT(9) NOT NULL, -- Número de cuenta de nuestro usuario
    PRIMARY KEY (id) -- Definimos la clave primaria como 'id'
);

INSERT INTO users (name, email, universidad, cuenta) VALUES ('YAHIR ADRIAN RUIZ MACHUCA', 'yahiradrianruiz10@gmail.com.mx', 'UNAM', '319123456');
INSERT INTO users (name, email, universidad, cuenta) VALUES ('DIANA JARAMILLO LOPEZ PALACIOS', 'dianalopez27@outlook.com.mx', 'C.U (FACULTAD DE INGENIERIA)', '319168746');

SELECT * FROM users;

```

Result Grid:

	id	name	email	universidad	cuenta
1	1	YAHIR ADRIAN RUIZ MACHUCA	yahiradrianruiz10@gmail.com.mx	UNAM	319123456
2	3	DIANA JARAMILLO LOPEZ PALACIOS	dianalopez27@outlook.com.mx	C.U (FACULTAD DE INGENIERIA)	319168746
3	4				

Action Output:

#	Time	Action	Message	Duration / Fetch
8	14:25:17	SELECT * FROM users LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
9	15:46:36	SELECT * FROM users LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
10	16:15:42	SELECT * FROM users LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

Código SQL para crear la Base de Datos crud en referencia a las acciones que realiza dicha página web.

```
bd.sql
```

```
1 -- Creación de BD
2 CREATE DATABASE IF NOT EXISTS crud;
3 USE crud;
4
5 -- Creación de tabla 'users'
6 CREATE TABLE IF NOT EXISTS users (
7     id INT AUTO_INCREMENT NOT NULL,
8     name VARCHAR(70) NOT NULL,
9     email VARCHAR(180) NOT NULL,
10    universidad VARCHAR(100) NOT NULL,
11    cuenta INT(9) NOT NULL,
12    PRIMARY KEY (id)
13 );
14
15 CREATE USER IF NOT EXISTS 'root'@'localhost' IDENTIFIED BY '123456';
16
17 GRANT ALL PRIVILEGES ON crud.* TO 'root'@'localhost';
18
19
20
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

Conexión lograda desde http://localhost:2345  
Conectado exitosamente a la Base de Datos crud

Lín. 20, col. 1 Espacios: 4 UTF-8 CRLF MS SQL Go Live Ninja 04:23 p. m. 27/11/2024

Este código en **Node.js** utiliza el framework **Express.js** para configurar un enrutador que define varias rutas específicas del servidor.

Se importa el módulo **express** y se crea una instancia de un enrutador con **express.Router()**. Además, se utiliza **path** para manejar rutas de archivos de forma compatible con diferentes sistemas operativos.

1. **Rutas para renderizar vistas:** La **ruta/edit** renderiza la vista **edit.ejs** ubicada en la carpeta **views**. La **ruta/agregar** renderiza la vista **add.ejs** en la misma carpeta. Esto permite mostrar páginas dinámicas generadas en el servidor al cliente.
2. **Rutas para servir archivos CSS:** Las rutas **/styles**, **/estilosagregar** y **/estiloseditar** envían al cliente los archivos CSS (**estilos.css**, **estilosadd.css**, **estilosedit.css**) almacenados en la carpeta **public/css**. Esto permite aplicar estilos específicos a las vistas en función de la página solicitada.
3. **Exportación del enrutador:** Finalmente, el enrutador se exporta con **module.exports = router** para que pueda ser integrado en el servidor principal.

En resumen, este código organiza las rutas del servidor para gestionar tanto vistas dinámicas como la entrega de recursos estáticos de manera eficiente.

```

const express = require("express");
const path = require("path");
const router = express.Router();

router.get('/edit', (req, res) =>{
    res.render(path.join(__dirname, '../views', 'edit.ejs'));
});

router.get('/agregar', (req, res) =>{
    res.render(path.join(__dirname, '../views', 'add.ejs'));
});

router.get('/estilos', (req, res) => {
    res.sendFile(path.join(__dirname,'../public/css','estilos.css'));
});

router.get('/estilosadd', (req, res) => {
    res.sendFile(path.join(__dirname,'../public/css','estilosadd.css'));
});

router.get('/estilosedit', (req, res) => {
    res.sendFile(path.join(__dirname,'../public/css','estilosedit.css'));
});

module.exports = router;

```

TERMINAL

```

Conexión lograda desde http://localhost:2345
Conectado exitosamente a la Base de Datos crud

```

Este código es una aplicación en Node.js con Express.js que implementa un sistema CRUD (Crear, Leer, Actualizar y Eliminar) para gestionar usuarios desde una base de datos MySQL.

Las funcionalidades principales incluyen:

- Operaciones CRUD:
  - a) Mostrar: Obtiene todos los usuarios de la tabla users y los muestra en una vista llamada index.ejs.
  - b) Agregar: Crea un nuevo usuario verificando que el campo "cuenta" tenga 9 dígitos.
  - c) Editar: Muestra un formulario de edición para un usuario específico y actualiza su información en la base de datos.
  - d) Eliminar: Borra un usuario específico basado en su ID.
- Inicio del servidor: Ejecuta el servidor en el puerto 2345 y muestra un mensaje de confirmación en la consola si es que se logra la conexión; si no es el caso se muestra un mensaje de error en la conexión.

```
const express = require('express');
const mysql = require('mysql2');
const bodyParser = require('body-parser');
const path = require('path');
const pageRoutes = require('./routes/pages');

const app = express();
const port = 2345;

app.use('/', pageRoutes);
app.use(express.static(path.join(__dirname, 'public')));

app.use(bodyParser.urlencoded({ extended: false }));
app.set('view engine', 'ejs');

// Conexión a la BD
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '123456',
  database: 'crud',
  port: '3306'
});

db.connect(err => {
  if(err)
    console.log('No se pudo realizar la conexión a la Base de Datos');
  else
    console.log('Conectado exitosamente a la Base de Datos crud');
})
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

Conexión lograda desde http://localhost:2345  
Conectado exitosamente a la Base de Datos crud

```
db.connect(err => {
  if(err)
    console.log('No se pudo realizar la conexión a la Base de Datos');
  else
    console.log('Conectado exitosamente a la Base de Datos crud');

  // Inicia servidor
  app.listen(port, ()=>{
    console.log(`Conexión lograda desde http://localhost:${port}`);
  });
});

// Mostrar tabla de usuarios
app.get('/', (req, res) => {
  const query = 'SELECT * FROM users';
  db.query(query, (err, results) => {
    if(err){
      console.error('Error al recuperar los datos. Código de error: ${err}');
      res.send('Error al recuperar los datos');
    } else{
      res.render('index', {users: results});
    }
  });
});

// Agregar un usuario
app.post('/add', (req, res) => {
  const { name, email, universidad, cuenta } = req.body;

  // Validamos que la cuenta tenga 9 dígitos
  if (!/^d{9}$/.test(cuenta))
    return res.send('El campo "Cuenta" debe contener 9 dígitos.');
})
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

Conexión lograda desde http://localhost:2345  
Conectado exitosamente a la Base de Datos crud

```
PROYECTO_FINAL
bd.sql pages.js app.js index.js add.js edit.js index.css add.css edit.css
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
Conexión lograda desde http://localhost:2345
Conectado exitosamente a la Base de Datos crud
Un. 122, col. 1 Espacios: 4 UTF-8 CRLF {} JavaScript Go Live Ninja 04:27 p. m. 27/11/2024
```

```
app.post('/add', (req, res) => {
  // Insertamos un nuevo usuario en la BD
  const query = "INSERT INTO users (name, email, universidad, cuenta) VALUES (?, ?, ?, ?)";
  db.query(query, [name, email, universidad, cuenta], (err) => {
    if (err) {
      console.error("Error al insertar datos en la tabla de usuarios. Código de error: ${err}");
      res.send("Error al insertar datos en la tabla de usuarios.");
    } else {
      res.redirect('/');
    }
  });
}

// Mostramos el formulario de la edición de un usuario
app.get('/edit/:id', (req, res) => {
  const { id } = req.params;
  const query = 'SELECT * FROM users WHERE id = ?'; // Buscamos al usuario con el ID especificado
  db.query(query, [id], (err, results) => {
    if(err){
      console.error('Error en la BD');
      res.send('Error en la BD');
    } else{
      res.render('edit', {users: results[0]}); // Mostramos el formulario con los datos del usuario
    }
  });
}

// Editar datos de usuario
app.post('/edit/:id', (req, res) => {
  const { id } = req.params;
  const { name, email, universidad, cuenta } = req.body;
```

```
PROYECTO_FINAL
bd.sql pages.js app.js index.js add.js edit.js index.css add.css edit.css
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
Un. 122, col. 1 Espacios: 4 UTF-8 CRLF {} JavaScript Go Live Ninja 04:27 p. m. 27/11/2024
```

```
app.post('/edit/:id', (req, res) => {
  // Validamos que la cuenta tenga 9 dígitos
  if (!/^d{9}$/.test(cuenta)) {
    return res.send('El campo "Cuenta" debe contener 9 dígitos.');
  }

  // Actualizamos los datos del usuario en la BD
  const query = 'UPDATE users SET name = ?, email = ?, universidad = ?, cuenta = ? WHERE id = ?';
  db.query(query, [name, email, universidad, cuenta, id], (err, results) => {
    if(err){
      console.error('Error en la BD');
      res.send('Error en la BD');
    } else{
      res.redirect('/');
    }
  });
}

// Eliminar un usuario
app.get('/delete/:id', (req, res) => {
  const { id } = req.params;
  const query = 'DELETE FROM users WHERE id = ?';
  db.query(query, [id], (err) => {
    if(err){
      console.error('Error al Eliminar');
      res.send('Error al Eliminar');
    } else{
      res.redirect('/');
    }
  });
});
```

Este código HTML dinámico con **EJS** muestra un registro de usuarios conectado a una base de datos. Incluye:

- Botón para **agregar** usuarios (agregar).
- Una tabla que lista usuarios (ID, Nombre, Email, Universidad, Cuenta) con datos generados dinámicamente.
- Botones para **editar** (edit) y **eliminar** (delete) cada usuario.
- Scripts (particles.js) para animar el fondo.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="/css/index.css">
  <title>CRUD (Proyecto Final)</title>
</head>
<body>
  <div id="particles-js"></div>
  <div class="centrar">
    <div class="borde">
      <div class="header">
        <h1>Proyecto Final (CRUD)</h1>
        <a href="/agregar"><button class="agregar">Agregar</button></a>
      </div>
      <table>
        <thead>
          <tr>
            <th>ID</th>
            <th>Nombre</th>
            <th>Correo</th>
            <th>Universidad</th>
            <th>Cuenta</th>
            <th>Opciones</th>
          </tr>
        </thead>
        <tbody>
          % users.forEach(user => { %
            <tr>
              <td class="body">% user.id %</td>
              <td class="body">% user.name %</td>
              <td class="body">% user.email %</td>
              <td class="body">% user.universidad %</td>
              <td class="body">% user.cuenta %</td>
              <td class="acciones">
                <a href="/edit/% user.id %"><button class="editar">Editar</button></a> | 
                <a href="/delete/% user.id %"><button class="eliminar">Eliminar</button></a>
              </td>
            </tr>
          % });
        </tbody>
      </table>
    </div>
  </div>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
  <body>
    <div class="centrar">
      <div class="borde">
        <div class="header">
          <h1>Proyecto Final (CRUD)</h1>
          <a href="/agregar"><button class="agregar">Agregar</button></a>
        </div>
        <table>
          <thead>
            <tr>
              <th>ID</th>
              <th>Nombre</th>
              <th>Correo</th>
              <th>Universidad</th>
              <th>Cuenta</th>
              <th>Acciones</th>
            </tr>
          </thead>
          <tbody>
            <tr>
              <td class="body">% user.id %</td>
              <td class="body">% user.name %</td>
              <td class="body">% user.email %</td>
              <td class="body">% user.universidad %</td>
              <td class="body">% user.cuenta %</td>
              <td class="acciones">
                <a href="/edit/% user.id %"><button class="editar">Editar</button></a> | 
                <a href="/delete/% user.id %"><button class="eliminar">Eliminar</button></a>
              </td>
            </tr>
          </tbody>
        </table>
      </div>
    </div>
  </body>
</html>

```

## Este código agrega usuarios a la Base de Datos.

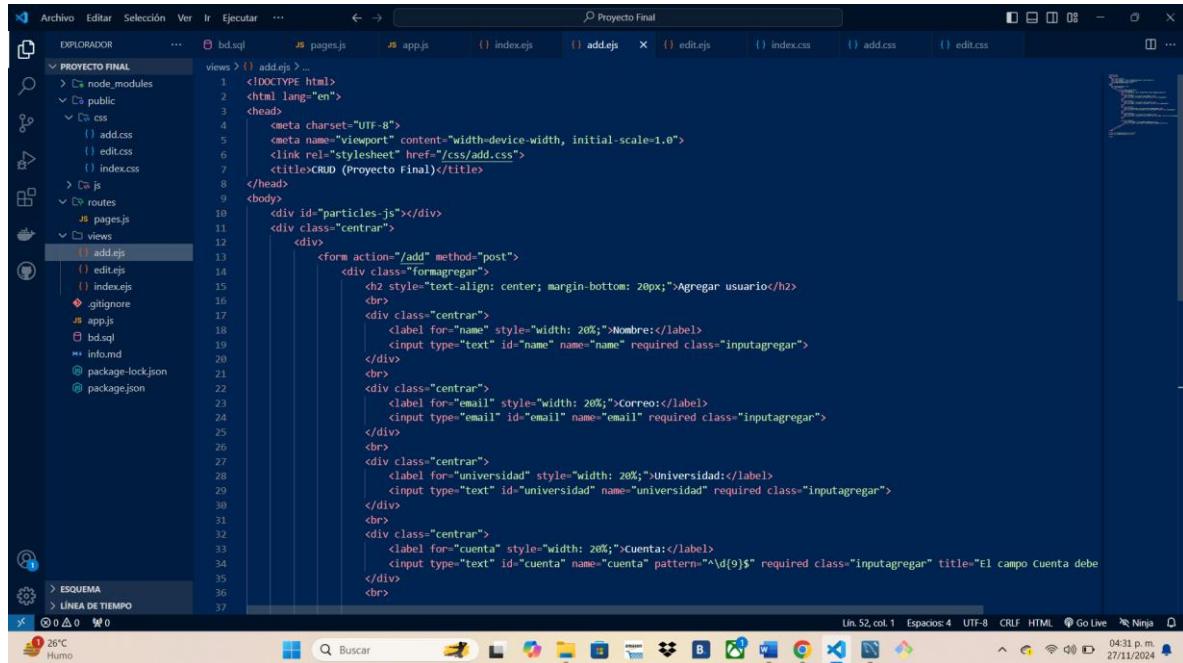
Permite enviar un formulario con datos del usuario (nombre, correo, universidad y cuenta) al servidor mediante una solicitud POST al endpoint /add. La cuenta debe contener exactamente 9 dígitos (validado con pattern).

### Componentes principales:

- Formulario de usuario: Campos de entrada para nombre, correo, universidad y cuenta con validación básica.
- Botones: Uno para enviar los datos al servidor y otro para cancelar (redirige a la página principal).

### Estilo y scripts:

- Archivo CSS (agregar.css) para el diseño.
- Animaciones de fondo con particles.js.



The screenshot shows the Visual Studio Code interface with the 'PROYECTO FINAL' folder open. The 'views' folder contains several files: add.ejs, edit.ejs, index.ejs, and add.css. The 'add.ejs' file is selected and shown in the editor pane. The code in 'add.ejs' is a JSP file with the following content:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/add.css">
    <title>CRUD (Proyecto final)</title>
</head>
<body>
    <div id="particles-js"></div>
    <div class="centrar">
        <form action="/add" method="post">
            <div class="formagregar">
                <h2 style="text-align: center; margin-bottom: 20px;">Añadir usuario</h2>
                <br>
                <div class="centrar">
                    <label for="name" style="width: 20%;>Nombre:</label>
                    <input type="text" id="name" name="name" required class="inputagregar">
                </div>
                <br>
                <div class="centrar">
                    <label for="email" style="width: 20%;>Correo:</label>
                    <input type="email" id="email" name="email" required class="inputagregar">
                </div>
                <br>
                <div class="centrar">
                    <label for="universidad" style="width: 20%;>Universidad:</label>
                    <input type="text" id="universidad" name="universidad" required class="inputagregar">
                </div>
                <br>
                <div class="centrar">
                    <label for="cuenta" style="width: 20%;>Cuenta:</label>
                    <input type="text" id="cuenta" name="cuenta" pattern="^\d{9}$" required class="inputagregar" title="El campo Cuenta debe contener exactamente 9 dígitos.">
                </div>
            <br>
        </div>
    </div>
</body>
```

```

views > add.ejs > ...
  2   <html lang="en">
  3     <body>
  4       <div class="centrar">
  5         <div>
  6           <form action="/edit" method="post">
  7             <div>
  8               <label for="cuenta" style="width: 20%;>Cuenta:</label>
  9               <input type="text" id="cuenta" name="cuenta" pattern="^\d{9}$" required class="inputagregar" title="El campo Cuenta debe tener exactamente 9 dígitos.">
  10              <br>
  11            <div class="centrar">
  12              <label for="cuenta" style="width: 20%;>Cuenta:</label>
  13              <input type="text" id="cuenta" name="cuenta" pattern="^\d{9}$" required class="inputagregar" title="El campo Cuenta debe tener exactamente 9 dígitos.">
  14            </div>
  15          </div>
  16        </div>
  17      </div>
  18    </div>
  19  </div>
  20  </div>
  21  </div>
  22  </div>
  23  </div>
  24  </div>
  25  </div>
  26  </div>
  27  </div>
  28  </div>
  29  </div>
  30  </div>
  31  </div>
  32  </div>
  33  </div>
  34  </div>
  35  </div>
  36  </div>
  37  </div>
  38  </div>
  39  </div>
  40  </div>
  41  </div>
  42  </div>
  43  </div>
  44  </div>
  45  </div>
  46  </div>
  47  </div>
  48  <script src="/js/particles.min.js"></script>
  49  <script src="/js/particulas.js"></script>
  50
  51 </html>

```

Este código **edita los datos de un usuario** en la Base de Datos.

Básicamente modifica los datos de un usuario específico mediante un formulario que realiza una solicitud POST al **endpoint /edit:id**, donde **:id** es el identificador del usuario. Los campos incluyen validaciones, como que el número de cuenta debe tener exactamente 9 dígitos.

### Componentes principales:

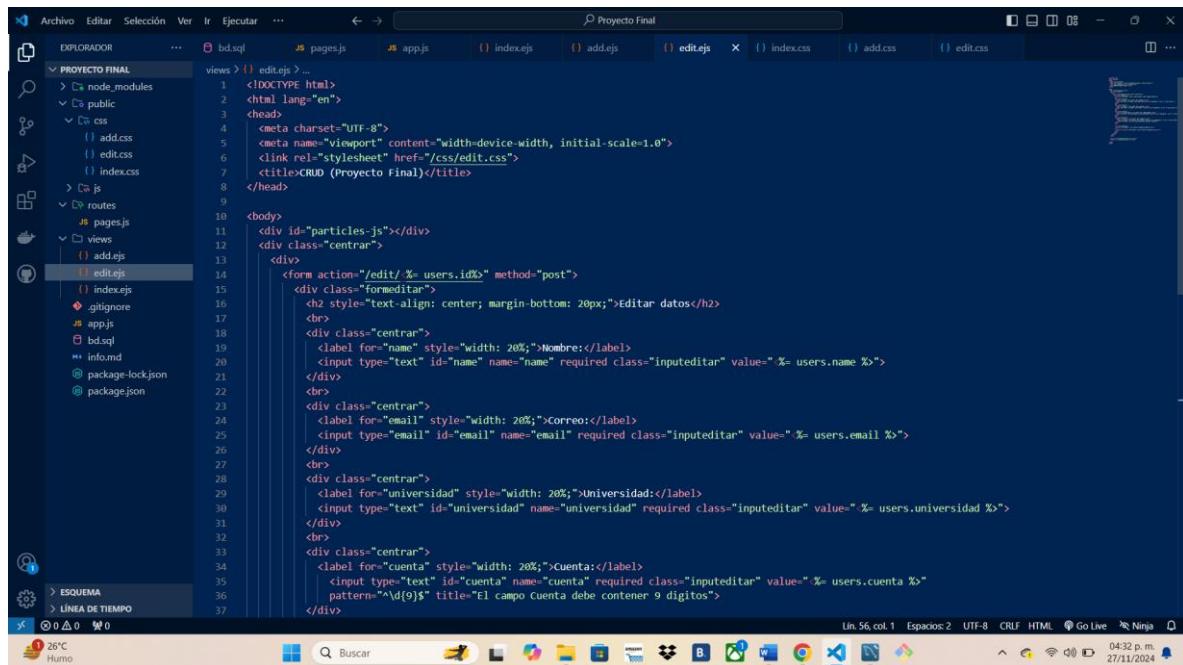
- Formulario de edición:
- Campos pre-rellenados con la información del usuario (name, email, universidad, cuenta) utilizando EJS para pasar datos desde el servidor.
- Validaciones incluidas (pattern) para asegurar que los datos sean correctos.

### Botones:

- Actualizar: Envía los datos actualizados al servidor.
- Cancelar: Redirige a la página principal sin realizar cambios.

### Estilo y scripts:

- Archivo CSS (editar.css) para personalizar el diseño de la página.
- Animaciones de fondo con particles.js.



Archivo Editar Selección Ver Ir Ejecutar ... 🔍 Proyecto Final

EXPLORADOR PROYECTO\_FINAL

views > edit.ejs > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/edit.css">
    <title>CRUD (Proyecto Final)</title>
```

body>

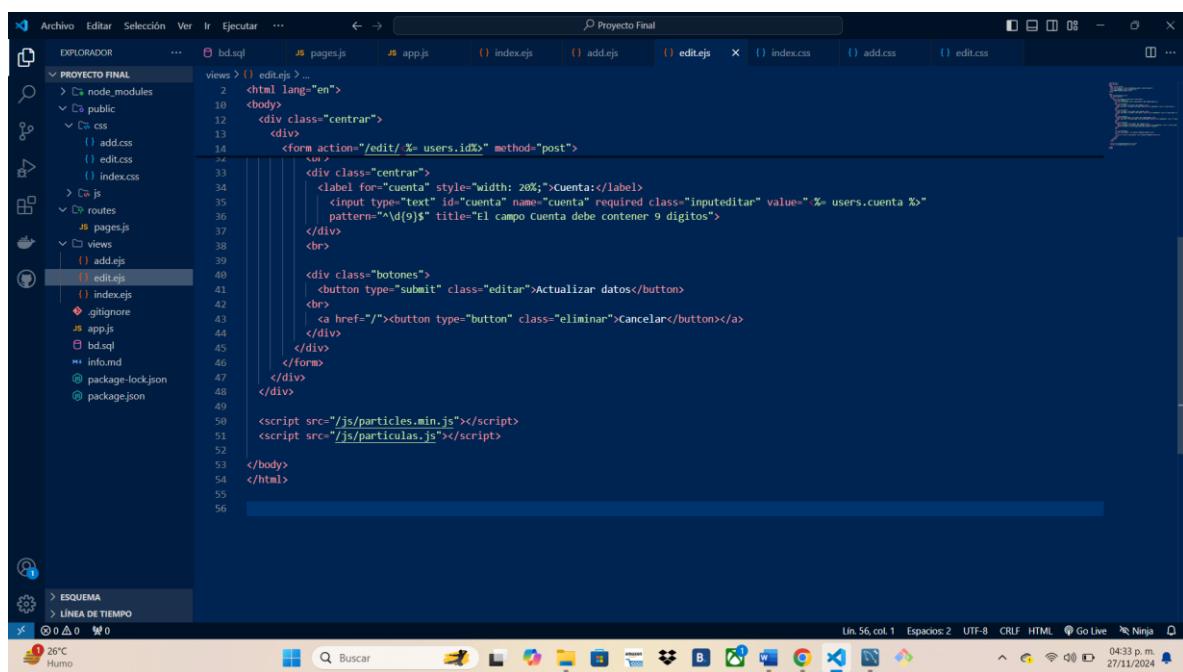
```
<div id="particles-js"></div>
<div class="centrar">
    <div action="/edit/%-users.id%" method="post">
        <div class="formeditar">
            <h2 style="text-align: center; margin-bottom: 20px;">Editar datos</h2>
            <br>
            <div class="centrar">
                <label for="name" style="width: 20%;>Nombre:</label>
                <input type="text" id="name" name="name" required class="inputeditar" value="%- users.name %">
            </div>
            <br>
            <div class="centrar">
                <label for="email" style="width: 20%;>correo:</label>
                <input type="email" id="email" name="email" required class="inputeditar" value="%- users.email %">
            </div>
            <br>
            <div class="centrar">
                <label for="universidad" style="width: 20%;>Universidad:</label>
                <input type="text" id="universidad" name="universidad" required class="inputeditar" value="%- users.universidad %">
            </div>
            <br>
            <div class="centrar">
                <label for="cuenta" style="width: 20%;>Cuenta:</label>
                <input type="text" id="cuenta" name="cuenta" required class="inputeditar" value="%- users.cuenta %"
                    pattern="\d{9}" title="El campo Cuenta debe contener 9 dígitos">
            </div>
        </div>
    </div>
</div>
```

LÍNEA DE TIEMPO

ESQUEMA

Lin. 56, col. 1 Espacios: 2 UTF-8 CRLF HTML Go Live Ninja 04:32 p. m. 27/11/2024

26°C Humo



Archivo Editar Selección Ver Ir Ejecutar ... 🔍 Proyecto Final

EXPLORADOR PROYECTO\_FINAL

views > edit.ejs > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/edit.css">
    <title>CRUD (Proyecto Final)</title>
```

body>

```
<div id="particles-js"></div>
<div class="centrar">
    <div action="/edit/%-users.id%" method="post">
        <div class="centrar">
            <label for="cuenta" style="width: 20%;>Cuenta:</label>
            <input type="text" id="cuenta" name="cuenta" required class="inputeditar" value="%- users.cuenta %"
                pattern="\d{9}" title="El campo Cuenta debe contener 9 dígitos">
        </div>
        <br>
        <div class="botones">
            <button type="submit" class="editar">Actualizar datos</button>
            <br>
            <a href="/"><button type="button" class="eliminar">Cancelar</button></a>
        </div>
    </div>
</div>
```

LÍNEA DE TIEMPO

ESQUEMA

Lin. 56, col. 1 Espacios: 2 UTF-8 CRLF HTML Go Live Ninja 04:33 p. m. 27/11/2024

26°C Humo

La hoja de CSS styles define animaciones y transiciones que mejoran la experiencia visual de la página, logrando efectos dinámicos y estéticos. Los elementos clave son:

### 1. Botones:

- Los botones tienen un fondo gris claro (--color-button) y texto negro (--color-text-button).
- Al pasar el cursor sobre ellos, el fondo cambia a un gris más oscuro (--color-button-hover) y se ajusta la opacidad.
- El botón "Agregar" está alineado a la derecha utilizando align-self: flex-end.

### 2. Tabla:

- La tabla tiene un fondo blanco, bordes colapsados y sombras suaves.
- Las celdas tienen padding y los encabezados están en gris claro con texto en mayúsculas.
- Las filas alternan entre blanco y gris claro, con un efecto de cambio de color al pasar el cursor (hover).

### 3. Página:

- El diseño de la página es limpio y moderno, con un fondo azul claro (--color-background).
- Los elementos están centrados y la estructura se organiza mediante un contenedor flex, con bordes redondeados y sombra para darle un efecto de profundidad.

```
public > css > () index.css > ...
1 :root {
2   --color-primary: #ffffff;
3   --color-secondary: #f5f5f5;
4   --color-accent: #00bbff; /* Azul para acciones */
5   --color-background: #eef2f7; /* Fondo azul claro */
6   --color-text: #333;
7   --color-border: #ddd;
8   --shadow: rgba(0, 0, 0, 0.1);
9   --color-button: #ccc; /* Gris claro para botones */
10  --color-button-hover: #b3b3b3; /* Gris más oscuro para hover */
11  --color-text-button: #000; /* Negro para el texto de los botones */
12 }
13
14 body {
15   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
16   font-size: 16px;
17   color: var(--color-text);
18   background-color: var(--color-background);
19   margin: 0;
20   padding: 20px;
21 }
22
23 .centrar {
24   max-width: 1000px;
25   margin: 20px auto;
26   padding: 20px;
27   background: var(--color-primary);
28   border: 1px solid var(--color-border);
29   border-radius: 10px;
30   box-shadow: 0 4px 8px var(--shadow);
31   display: flex;
32   flex-direction: column;
33   align-items: flex-start;
34 }
35
36 h1 {
37   font-size: 28px;
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "PROYECTO\_FINAL".
- Editor:** Displays the content of the `index.css` file.
- Bottom Status Bar:** Shows "Lin. 126, col. 1 Espacios: 4 UTF-8 CRLF CSS Go Live Ninja" and the date "27/11/2024".
- Taskbar:** Shows various application icons.

```
public > css > index.css > ...
35 h1 {
36   font-size: 28px;
37   margin-bottom: 20px;
38   color: var(--color-accent);
39   text-align: center;
40 }
41 button {
42   font-size: 16px;
43   padding: 10px 15px;
44   border: none;
45   border-radius: 5px;
46   cursor: pointer;
47   font-weight: bold;
48   transition: all 0.3s ease;
49   background: var(--color-button);
50   color: var(--color-text-button);
51 }
52 button:hover {
53   opacity: 0.9;
54   background: var(--color-button-hover);
55 }
56 .agregar {
57   width: fit-content;
58   align-self: flex-end; /* Alinea el botón a la derecha */
59 }
60 table {
61   width: 100%;
62   border-collapse: collapse;
63   background: var(--color-primary);
64   box-shadow: 0 2px 4px var(--shadow);
65   margin-top: 20px;
66   border-radius: 5px;
67 }
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "PROYECTO\_FINAL".
- Editor:** Displays the content of the `index.css` file, identical to the previous screenshot but with some changes highlighted in red.
- Bottom Status Bar:** Shows "Lin. 126, col. 1 Espacios: 4 UTF-8 CRLF CSS Go Live Ninja" and the date "27/11/2024".
- Taskbar:** Shows various application icons.

```
public > css > index.css > ...
65 table {
66   overflow: hidden;
67 }
68 th, td {
69   padding: 12px 15px;
70   text-align: left;
71   border-bottom: 1px solid var(--color-border);
72 }
73 th {
74   background-color: var(--color-secondary);
75   font-weight: bold;
76   color: var(--color-text);
77   text-transform: uppercase;
78   font-size: 14px;
79 }
80 td {
81   font-size: 14px;
82   color: var(--color-text);
83 }
84 tr:nth-child(even) {
85   background: #f9f9f9;
86 }
87 tr:hover {
88   background: rgba(0, 123, 255, 0.1);
89   transition: background 0.3s ease;
90 }
91 .acciones {
92   display: flex;
93   gap: 10px;
94   justify-content: center;
95 }
```

The screenshot shows a code editor interface with the title bar "Proyecto Final". The left sidebar displays a file tree for a project named "PROYECTO\_FINAL". The "index.css" file is selected and open in the main editor area. The CSS code defines styles for buttons, a footer, and a form element.

```
public > css > index.css > ...
108 .editar {
109   background: var(--color-button);
110   color: var(--color-text-button);
111 }
112 }
113
114 .eliminar {
115   background: var(--color-button);
116   color: var(--color-text-button);
117 }
118
119 footer {
120   margin-top: 20px;
121   text-align: center;
122   font-size: 14px;
123   color: var(--color-text);
124   opacity: 0.8;
125 }
126
```

The status bar at the bottom shows "Lin. 126, col. 1 Espacios: 4 UTF-8 CRLF CSS Go Live Ninja 04:38 p. m. 27/11/2024".

Este código en CSS corresponde al diseño de mi sistema para agregar usuarios a la Base de Datos.

The screenshot shows a code editor interface with the title bar "Proyecto Final". The left sidebar displays a file tree for a project named "PROYECTO\_FINAL". The "add.css" file is selected and open in the main editor area. The CSS code defines styles for the root element, body, a central container, and an h1 element.

```
public > css > add.css > ...
1 :root {
2   --color-primary: #ffffff;
3   --color-secondary: #ff5f5f;
4   --color-background: #e8f5e9; /* Fondo claro */
5   --color-text: #0000;
6   --color-border: #ddd;
7   --shadow: rgba(0, 0, 0, 0.1);
8 }
9
10 body {
11   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
12   font-size: 16px;
13   color: var(--color-text);
14   background-color: var(--color-background);
15   margin: 0;
16   padding: 20px;
17 }
18
19 .centrar {
20   max-width: 600px;
21   margin: 20px auto;
22   padding: 20px;
23   background: var(--color-primary);
24   border: 1px solid var(--color-border);
25   border-radius: 10px;
26   box-shadow: 0 4px 8px var(--shadow);
27 }
28
29 h1 {
30   font-size: 24px;
31   margin-bottom: 20px;
32   color: var(--color-accent);
33   text-align: center;
34 }
35
36 form {
37   display: flex;
```

The status bar at the bottom shows "Lin. 102, col. 1 Espacios: 4 UTF-8 CRLF CSS Go Live Ninja 04:43 p. m. 27/11/2024".

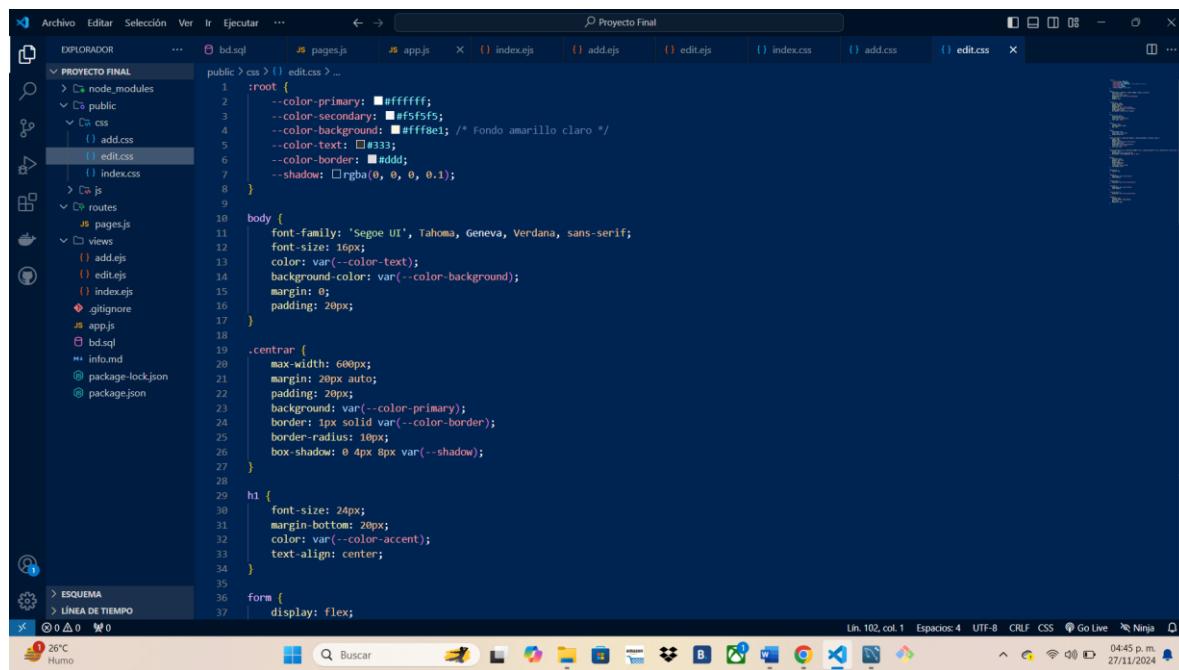
```
public > css > add.css > ...
36 form {
37   flex-direction: column;
38   gap: 15px;
39 }
40
41 label {
42   font-weight: bold;
43   color: var(--color-text);
44   margin-bottom: 5px;
45 }
46
47 input[type="text"], input[type="number"], input[type="email"], textarea, select {
48   width: 100%;
49   padding: 10px;
50   border: 1px solid var(--color-border);
51   border-radius: 5px;
52   background: var(--color-secondary);
53   color: var(--color-text);
54   font-size: 14px;
55 }
56
57 input[type="text"]:focus, input[type="number"]:focus, input[type="email"]:focus, textarea:focus, select:focus {
58   outline: none;
59   border-color: var(--color-accent);
60   box-shadow: 0 0 0 5px rgba(76, 175, 80, 0.5);
61 }
62
63 button {
64   font-size: 16px;
65   padding: 10px 15px;
66   border: none;
67   border-radius: 5px;
68   cursor: pointer;
69   transition: all 0.3s ease;
70   font-weight: bold;
71 }
```

Lineas de código: 73, Col. 1, Espacio: 4, UTF-8, CRLF, CSS, Go Live, Ninja, 04:43 p.m., 27/11/2024

```
public > css > add.css > ...
73 button:hover {
74   opacity: 0.9;
75 }
76
77 .guardar {
78   background: var(--color-accent);
79   color: white;
80 }
81
82 .guardar:hover {
83   background: var(--color-accent-hover);
84 }
85
86 .cancelar {
87   background: var(--color-cancel);
88   color: white;
89 }
90
91 .cancelar:hover {
92   background: var(--color-cancel-hover);
93 }
94
95 .botones {
96   display: flex;
97   justify-content: space-between;
98   gap: 15px;
99   margin-top: 20px;
100 }
```

Lineas de código: 102, Col. 1, Espacio: 4, UTF-8, CRLF, CSS, Go Live, Ninja, 04:43 p.m., 27/11/2024

Este CSS corresponde a la interfaz gráfica para visualizar el sistema de actualización o edición de datos por parte del usuario.



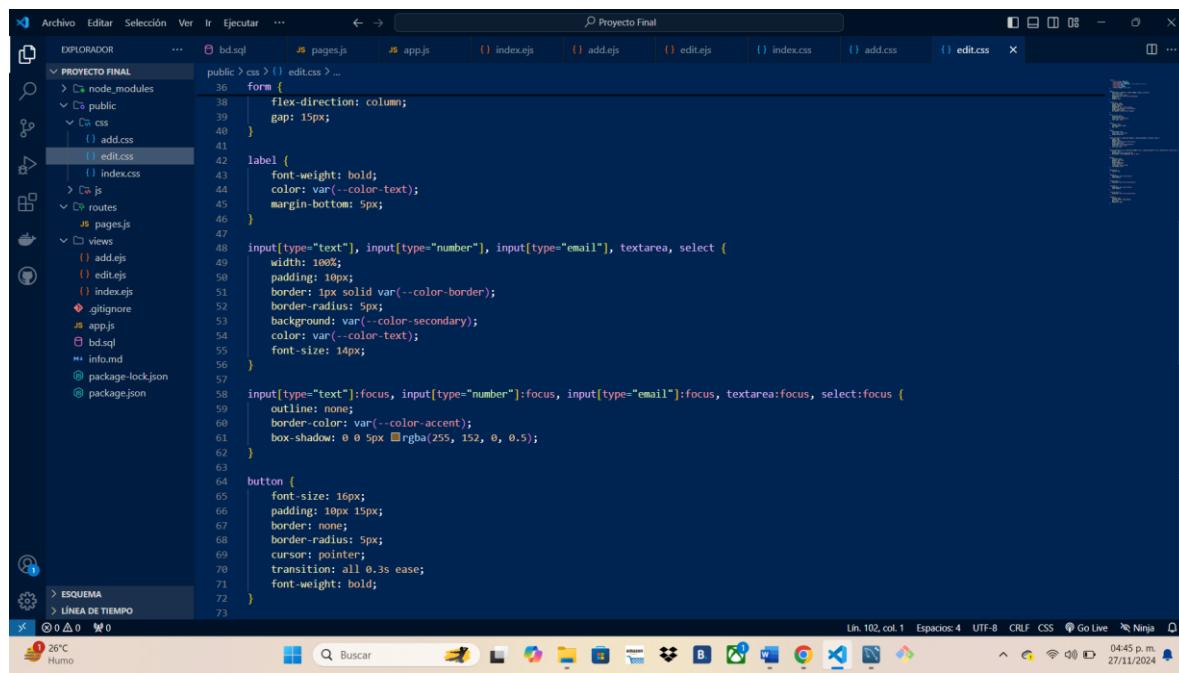
```
public > css > edit.css ...
root {
    --color-primary: #ffffff;
    --color-secondary: #f5f5f5;
    --color-background: #ffffe1; /* Fondo amarillo claro */
    --color-text: #333;
    --color-border: #ddd;
    --shadow: rgba(0, 0, 0, 0.1);
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    font-size: 16px;
    color: var(--color-text);
    background-color: var(--color-background);
    margin: 0;
    padding: 20px;
}

.centrar {
    max-width: 600px;
    margin: 20px auto;
    padding: 20px;
    background: var(--color-primary);
    border: 1px solid var(--color-border);
    border-radius: 10px;
    box-shadow: 0 4px 8px var(--shadow);
}

h1 {
    font-size: 24px;
    margin-bottom: 20px;
    color: var(--color-accent);
    text-align: center;
}

form {
    display: flex;
}
```



```
form {
    flex-direction: column;
    gap: 15px;
}

label {
    font-weight: bold;
    color: var(--color-text);
    margin-bottom: 5px;
}

input[type="text"], input[type="number"], input[type="email"], textarea, select {
    width: 100%;
    padding: 10px;
    border: 1px solid var(--color-border);
    border-radius: 5px;
    background: var(--color-secondary);
    color: var(--color-text);
    font-size: 14px;
}

input[type="text"]:focus, input[type="number"]:focus, input[type="email"]:focus, textarea:focus, select:focus {
    outline: none;
    border-color: var(--color-accent);
    box-shadow: 0 0 5px rgba(255, 152, 0, 0.5);
}

button {
    font-size: 16px;
    padding: 10px 15px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: all 0.3s ease;
    font-weight: bold;
}
```

```
public > css > edit.css > ...
73
74   button:hover {
75     |   opacity: 0.9;
76   }
77
78   .guardar {
79     |   background: var(--color-accent);
80     |   color: white;
81   }
82
83   .guardar:hover {
84     |   background: var(--color-accent-hover);
85   }
86
87   .cancelar {
88     |   background: var(--color-cancel);
89     |   color: white;
90   }
91
92   .cancelar:hover {
93     |   background: var(--color-cancel-hover);
94   }
95
96   .botones {
97     |   display: flex;
98     |   justify-content: space-between;
99     |   gap: 15px;
100    |   margin-top: 20px;
101  }
102
```

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a project structure named "PROYECTO\_FINAL" containing files like bd.sql, pages.js, app.js, index.ejs, add.ejs, edit.ejs, index.css, add.css, edit.css, routes.js, views.js, and info.md.
- Code Editor:** Displays a CSS file named "edit.css" with the following content:

```
public > css > edit.css > ...
73
74   button:hover {
75     |   opacity: 0.9;
76   }
77
78   .guardar {
79     |   background: var(--color-accent);
80     |   color: white;
81   }
82
83   .guardar:hover {
84     |   background: var(--color-accent-hover);
85   }
86
87   .cancelar {
88     |   background: var(--color-cancel);
89     |   color: white;
90   }
91
92   .cancelar:hover {
93     |   background: var(--color-cancel-hover);
94   }
95
96   .botones {
97     |   display: flex;
98     |   justify-content: space-between;
99     |   gap: 15px;
100    |   margin-top: 20px;
101  }
102
```
- Status Bar:** Shows "Lin. 102, col. 1 Espacios: 4 UTF-8 CRLF CSS Go Live Ninja" and a timestamp "27/11/2024".
- Taskbar:** Shows various application icons including File Explorer, Task Manager, and several browser windows.